This notebook contains simple examples for the `DataFrames.jl` join operations.

The different join operations can be roughly summarized as follows:

- `outerjoin`: `df1 OR df2`
- `innerjoin`: `df1 AND df2`
- `semijoin`: `df1 AND df2`, with only the columns from `df1`
- `antijoin`: `df1 AND !df2`, with only the columns from `df1`
- `leftjoin`: `df1`
- `rightjoin`: `df2`
- `crossjoin`: Cartesian product of rows from `df1` and `df2`

```
using DataFrames
```

df1 =

	A	B
1	1	"x"
2	2	"y"
3	3	"z"

```
df1 = DataFrame(A = 1:3, B = ["x", "y", "z"])
```

df2 =

	A	C
1	2	"xx"
2	3	"yy"
3	4	"zz"

```
df2 = DataFrame(A = 2:4, C = ["xx", "yy", "zz"])
```

	A	B	C
1	2	"y"	"xx"
2	3	"z"	"yy"
3	1	"x"	missing
4	4	missing	"zz"

```
# Include rows with values of A shared by df1 OR df2
outerjoin(df1, df2, on = :A)
```

	A	B	C
1	2	"y"	"xx"
2	3	"z"	"yy"

```
# Include rows with values of A shared by df1 AND df2
innerjoin(df1, df2, on = :A)
```

	A	B
1	2	"y"
2	3	"z"

```
# Same as innerjoin, except only keep columns from df1
semijoin(df1, df2, on = :A)
```

	A	B
1	1	"x"

```
# Include rows with values of A shared by df1 AND !df2.
# As in semijoin, only keep columns from df1.
antijoin(df1, df2, on = :A)
```

	A	B	C
1	1	"x"	missing
2	2	"y"	"xx"
3	3	"z"	"yy"

```
# Include rows with values of A shared by df1.
# Rows are ordered as they appear in df1.
leftjoin(df1, df2, on = :A, order = :left)
```

	A	B	C
1	2	"y"	"xx"
2	3	"z"	"yy"
3	4	missing	"zz"

```
# Include rows with values of A shared by df2.
rightjoin(df1, df2, on = :A)
```

	A	B	A_1	C
1	1	"x"	2	"xx"
2	1	"x"	3	"yy"
3	1	"x"	4	"zz"
4	2	"y"	2	"xx"
5	2	"y"	3	"yy"
6	2	"y"	4	"zz"
7	3	"z"	2	"xx"
8	3	"z"	3	"yy"
9	3	"z"	4	"zz"

```
# Include cartesian product of rows passed from all data frames.
# That is, if df1 has r1 rows, and df2 has r2 rows, the output
# data frame has r1 x r2 rows.
#
# Each of the m x n pairs of rows includes all columns from each
# data frame. E.g., if df1 has c1 columns and df2 has c2 columns,
# the output data frame has c1 + c2 columns.
#
# The `makeunique` parameter simply appends a suffix to conflicting
# column names so that column names in the output frame are unique.
crossjoin(df1, df2, makeunique = true)
```